Speeded Up Detection of Squared Fiducial Markers

Article *in* Image and Vision Computing · June 2018

DOI: 10.1016/j.imavis.2018.05.004

3 authors:

Francisco J. Romero-Ramirez
University of Cordoba (Spain)

Rafael Muñoz-Salinas
University of Cordoba (Spain)

Rafael Medina-Carnicer
University of Cordoba (Spain)

Speeded Up Detection of Squared Fiducial Markers

Francisco J. Romero-Ramirez[1], Rafael Muñoz-Salinas[1,2,*], Rafael Medina-Carnicer[1,2]

Abstract

Squared planar markers have become a popular method for pose estimation in applications such as autonomous robots, unmanned vehicles or virtual trainers. The markers allow estimating the position of a monocular camera with minimal cost, high robustness, and speed. One only needs to create markers with a regular printer, place them in the desired environment so as to cover the working area, and then registering their location from a set of images.

Nevertheless, marker detection is a time-consuming process, especially as the image dimensions grows. Modern cameras are able to acquire high resolutions images, but fiducial marker systems are not adapted in terms of computing speed. This paper proposes a multi-scale strategy for speeding up marker detection in video sequences by wisely selecting the most appropriate scale for detection, identification and corner estimation. The experiments conducted show that the proposed approach outperforms the state-of-the-art methods without sacrificing accuracy or robustness. Our method is up to 40 times faster than the state-of-the-art method, achieving over 1000 fps in 4K images without any parallelization.

Keywords: Fiducial Markers, Marker Mapping, SLAM.

1. Introduction

Pose estimation is a common task for many applications such as autonomous robots [1, 2, 3], unmanned vehicles [4, 5, 6, 7, 8] and virtual assistants [9, 10, 11, 12], among other.

Cameras are cheap sensors that can be effectively used for this task. In the ideal case, natural features such as keypoints or texture [13, 14, 15, 16] are be employed to create a map of the environment. Although some of the traditional problems of previous methods for this task have been solved in the last few years, other problems remain. For instance, they are subject to filter stability issues or significant computational requirements.

In any case, artificial landmarks are a popular approach for camera pose estimation. Square fiducial markers, comprised by an external squared black border and an internal identification code, are especially attractive because the camera pose can be estimated from the four corners of a single marker [17, 18, 19, 20]. The recent work of [21] is a step forward the use of this type of markers in large-scale problems. One only need to print the set of markers with a regular printer, place them in the area under which the camera must move, and take a set of pictures of the markers. The pictures are then analyzed and the three-dimensional marker locations automatically obtained. Afterward, a single image spotting a marker is enough to estimate the camera pose.

Despite the recent advances, marker detection can be a time-consuming process. Considering that the systems requiring localization have in many cases limited resources, such as mobile phones or aerial vehicles, the computational effort of localization should be kept to a minimum. The computing time employed in marker detection is a function of the image size employed: the larger the images, the slower the process. On the other hand, high-resolution images are preferable since markers can be detected, even if far from the camera, with high accuracy. The continuous reduction in the cost of the cameras, along with the increase of their resolution, makes necessary to develop methods able to reliably detect the markers in high-resolution images.

The main contribution of this paper is a novel method for detecting square fiducial markers in video sequences. The proposed method relies on the idea that markers can be detected in smaller versions of the image, and employs a multi-scale approach to speed up computation while maintaining the precision and accuracy. In addition, the system is able to dynamically adapt its parameters in order

*Corresponding author

Email addresses: fj.romero@uco.es (Francisco J. Romero-Ramirez), in1musar@uco.es (Rafael Muñoz-Salinas), rmedina@uco.es (Rafael Medina-Carnicer)

[1]Departamento de Informática y Análisis Numérico, Edificio Einstein. Campus de Rabanales, Universidad de Coŕdoba, 14071, Córdoba, Spain, Tlfn:(+34)957212289

[2]Instituto Maimónides de Investigación en Biomedicina (IMIBIC). Avenida Menéndez Pidal s/n, 14004, Córdoba, Spain, Tlfn:(+34)957213861

to achieve maximum performance in the analyzed video sequence. Our approach has been extensively tested and compared with the state-of-the-art methods for marker detection. The results show that our method is more than an order of magnitude faster than state-of-the-art approaches without compromising robustness or accuracy, and without requiring any type of parallelism.

The remainder of this paper is structured as follows. Section 2 explains the works most related to ours. Section 3 details our proposal for speeding up the detection of markers. Finally, Section 4 gives a exhaustive analysis of the proposed method and Section 5 draws some conclusions.

2. Related works

Fiducials marker systems are commonly used for camera localization and tracking when robustness, precision, and speed are required. In the simplest case, points are used as fiducial markers, such as LEDs, retroreflective spheres or planar dots [22, 23]. However, their main drawback is the need of a method to solve the assignment problem, i.e., assigning a unique and consistent identifier to each element over time. In order to ease the problem, a common solution consists in adding an identifying code into each marker. Examples of this are planar circular markers [24, 25], 2D-barcodes [26, 27] and even some authors have proposed markers designed using evolutionary algorithms [28].

Amongst all proposed approaches, these based on squared planar markers have gained popularity. These markers consist of an external black border and an internal code (most often binary) that uniquely identifies each marker (see Fig 1). Their main advantage is that the pose of the camera can be estimated from a single marker.

ARToolKit [29] is one of the pioneer proposals. They employed markers with a custom pattern that is identified by template matching. This identification method, however, is prone to error and not very robust to illumination changes. In addition, the method's sensitivity degrades as the number of markers increases. As a consequence, other authors improved that work by using binary BCH codes [30] (which allows a more robust error detection) and named it ARToolKit+ [31]. The project was halted and followed by the Studierstube Tracker project [32], which is privative. Similar to the ARToolKit+ project is the discontinued project ARTag [33].

BinARyID [34] is one of the first systems that proposed a method for generating customizable marker codes. Instead of using a predefined set of codes, they proposed a method for generating the desired number of codes for each particular application. However, they do not consider the possibility of error detection and correction. AprilTags [18], however, proposed methods for error detection and correction, but their approach was not suitable for a large number of markers.

The work ArUco [17] is probably the most popular system for marker detection nowadays. It adapts to non-uniform illumination, and is very robust, being able to do error detection and correction of the binary codes implemented. In addition, the authors proposed a method to obtain optimal binary codes (in terms of intermarker-distance) using Mixed Integer Linear Programming [35]. Chilitags [36] is a variation of ArUco that employs a simpler method for decoding the marker binary codes. As we show in the experimental section, the method has a bad behavior in high-resolution images.

The recent work [21] is a step towards the applicability of such methods to large areas, proposing a method for estimating the three-dimensional location of a set of markers freely placed in the environment (Fig 1). Given a set of images taken with a regular camera (such as a mobile phone), the method automatically estimates their location. This is an important step that allows extending the robust localization of fiducial markers to very large areas.

Although all fiducial marker systems aim maximum speed in their design, few specific solutions have been proposed to speed up the detection process. The work of Johnston *et. al.* [37] is an interesting example in which the authors propose a method to speed up computation by parallelizing the image segmentation process. Nevertheless, both speed and computing power is a crucial aspect, especially if the localization system needs to be embedded in devices with limited resources.

Our work can be seen as an improvement of the ArUco system, that according to our experience, is one of the most reliable fiducial marker systems nowadays (see Sec 4 for further details). We propose a novel method for marker detection and identification that allows to speed up the computing time in video sequences by wisely exploiting temporal information and an applying multi-scale approach. In contrast to previous works, no parallelization is required in our method, thus making it especially attractive for mobile devices with limited computational resources.

3. Speeded up marker detection

This section provides a detailed explanation of the method proposed for speeding up the detection of squared planar markers. First, Sect. 3.1 provides an overview of the pipeline employed in the previous work, ArUco [17], for marker detection and identification, highlighting the parts of the process susceptible to be accelerated. Then,

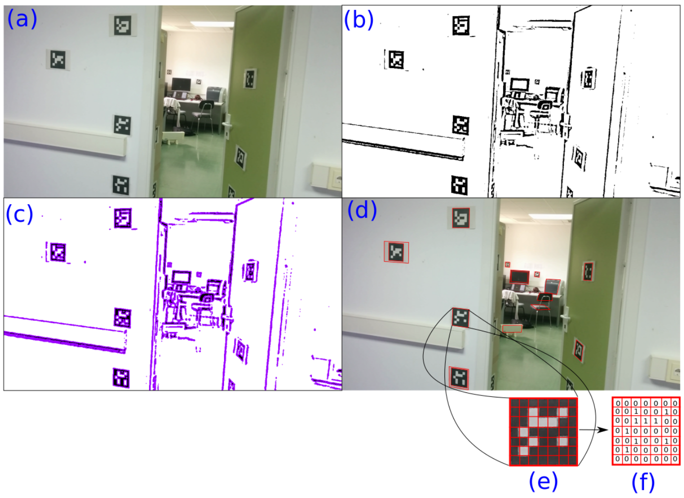


Figure 1: **Detection and identification pipeline of ArUco.** (a) Original image. (b) Image thresholded using an adaptive method. (c) Contours extracted. (d) Filtered contours that approximate to four-corner polygons. (e) Canonical image computed for one of the squared contours detected. (f) Binarization after applying Otsu's method.

Sect. 3.2 explains the proposed method to speed up the process.

3.1. Marker detection and identification in ArUco

The main steps for marker detection and identification proposed in ArUco [17] are depicted in Figure 1. Given the input image I (Figure 1a), the following steps are taken:

- Image segmentation (Figure 1b). Since the designed markers have an external black border surrounded by a white space, the borders can be found by segmentation. In their approach, a local adaptive method is employed: the mean intensity value m of each pixel is computed using a window size w_t. The pixel is set to zero if its intensity is greater than $m - c$, where c is a constant value. This method is robust and obtains good results for a wide range of values of its parameters w_t and c.

- Contour extraction and filtering (Figures 1(c,d)). The contour following algorithm of Suzuki and Abe [38] is employed to obtain the set of contours from the thresholded image. Since most of the contours extracted correspond to irrelevant background elements, a filtering step is required. First, contours too small are discarded. Second, the remaining contours are approximated to its most similar polygon using the Douglas and Peucker algorithm [39]. Those that do not approximate well to a four-corner convex polygon are discarded from further processing.

- Marker code extraction (Figures 1(e,f)). The next step consists in analyzing the inner region of the remaining contours to determine which of them are valid markers. To do so, perspective projection is first removed by computing the homography matrix, and the resulting canonical image (Fig. 1e) is thresholded using the Otsu's method [40]. The binarized image (Fig. 1f) is divided into a regular grid and each element is assigned a binary value according to the majority of the pixels in the cell. For each marker candidate, it is necessary to determine whether it belongs to the set of valid markers or if it is a background element. Four possible identifiers are obtained for each candidate, corresponding to the four possible rotations of the canonical image. If any of the identifiers belong to the set of valid markers, then it is accepted.

- Subpixel corner refinement. The last step consists in estimating the location of the corners with subpixel accuracy. To do so, the method employs a linear regression of the marker's contour pixels. In other words, it estimates the lines of the marker sides employing all the contour pixels and computes the intersections. This method, however, is not reliable for uncalibrated cameras with small focal lenses (such as fisheye cameras) since they usually exhibit high distortion.

When analyzing the computing times of this pipeline, it can be observed that the Image segmentation and the Marker code extraction steps are consuming most of the computing time. The time employed in the image segmentation step is proportional to the image size, that also influences the length of the contours extracted and thus the computing time employed in the Contour extraction and filtering step. The extraction of the canonical image (in the Marker code extraction step) involves two operations. First, computing the homography matrix, which is cheap. But then, the inner region of each contour must be warped to create the canonical image. This step requires access to the image pixels of the contour region performing an interpolation in order to obtain the canonical image. The main problem is that the time required to obtain the canonical image depends on the size of the observed contour. The larger a contour in the original image, the more time it is required to obtain the canonical image. Moreover, since most of the contours obtained do not belong to markers, the system may employ a large amount of time computing canonical images that will be later rejected.

A simpler approach to solving that problem would be to directly sample a few sets of pixels from the inner region of the marker. This is the method employed in ChiliTags. However, as it will be shown in the experimental section, it is prone to many false negatives.

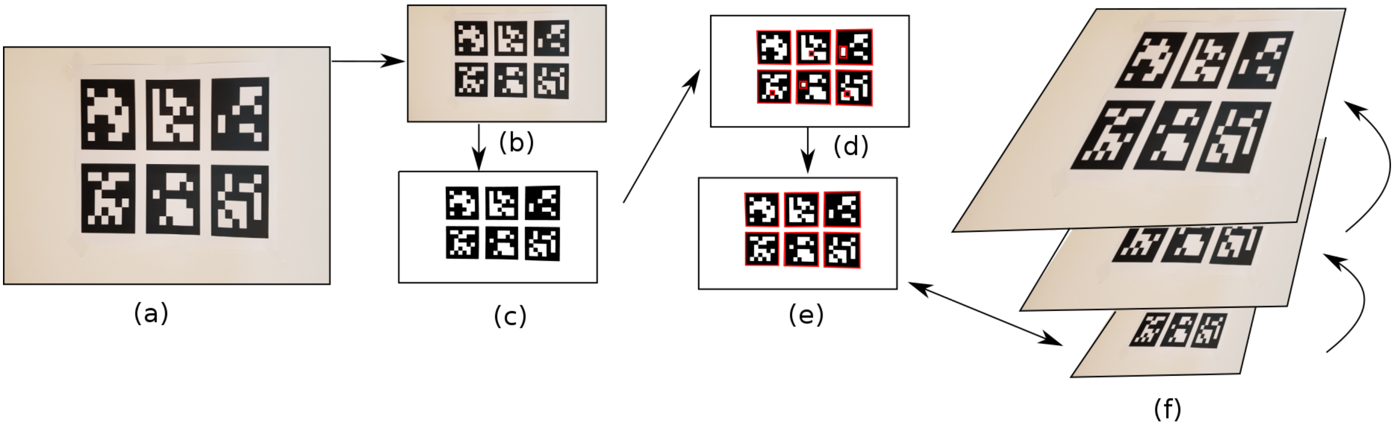


Figure 2: **Process pipeline** Main steps for fast detection and identification of squared planar markers.(a) Original input image. (b) Resized image for marker search. (c) Thresholded image. (d) Rectangles found (pink). (e) Markers detected with its corresponding identification. The image pyramid is used to speed up homography computation. (f) The corners obtained in (e) are upsampled to find their location in the original image with subpixel precision.

3.2. Proposed method

The key ideas of our proposal in order to speed up the computation are explained below. First, while the adaptive thresholding method employed in ArUco is robust to many illumination conditions without altering its parameters, it is a time-consuming process that requires a convolution. By taking advantage of temporal information, the adaptive thresholding method is replaced by a global thresholding approach.

Second, instead of using the original input image, a smaller version is employed. This is based on the fact that, in most cases, the useful markers for camera pose estimation must have a minimum size. Imagine an image of dimensions 1920×1080 pixels, in which a marker is detected as a small square with a side length of 10 pixels. Indeed, the estimation of the camera pose is not reliable at such small resolution. Thus, one might want to set a minimum length to the markers employed for camera pose estimation. For instance, let say that we only use markers with a minimum side length of $\dot{\tau}_i = 100$ pixels, i.e., with a total area of 10.000 pixels. Another situation in which we can set a limit to the length of markers is when processing video sequences. It is clear that the length of a marker must be similar to its length in the previous frame.

Now, let us also think about the size of the canonical images employed (Figure 1e). The smaller the image, the faster the detection process but the poorer the image quality. Our experience, however, indicates that very reliable detection of the binary code can be obtained from very small canonical images, such 32×32 pixels. In other words, all the rectangles detected in the image, no matter their side length, are reduced to canonical images of side length $\tau_c = 32$ pixels, for the purpose of identification.

Our idea, then, is to employ a reduced version of the input image, using the scale factor $\frac{\tau_c}{\dot{\tau}_i}$, so as to speed up the segmentation step. In the reduced image, the smallest allowed markers, with a side length of 100 pixels in the original image, appears as rectangles with a side length of 32 pixels. As a consequence, there will be no loss of quality when they are converted into the canonical image.

This idea has one drawback: the location of the corners extracted in the low resolution image is not as good estimations as the ones that can be obtained in the original image. Thus, the pose estimated with them will have a higher error. To solve that problem, a corner upsampling step is included, in which the precision of the corners is refined up to subpixel accuracy in the original input image by employing an image pyramid.

Finally, it must be considered that the generation of the canonical image is a very time-consuming operation (even if the process is done in the reduced image) that proportional to the contour length. We propose a method to perform the extraction of the canonical images in almost constant time (independently of the contour length) by wisely employing the image pyramid.

Below, there is a detailed explanation of the main steps of the proposed method, using Figure 2 to ease the explanation.

1. *Image Resize:* Given the input image I (Fig 2a), the first step consists in obtaining a resized version I^r (Fig 2b) that will be employed for segmentation. As previously pointed out, the size of the reduced image is calculated as:

$$I^r_w = \left\lfloor \frac{\tau_c}{\dot{\tau}_i} I_w \right\rfloor ; I^r_h = \left\lfloor \frac{\tau_c}{\dot{\tau}_i} I_h \right\rfloor , \quad (1)$$

where the subscripts $_w$ and $_h$ denotes width and height

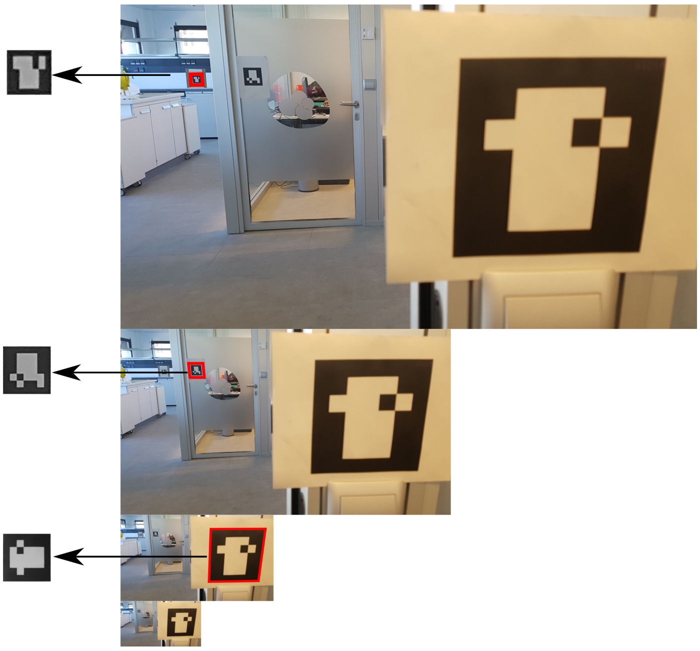


Figure 3: **Pyramidal Warping.** Scene showing tree marker at different resolutions. The left column shows the canonical images warped from the pyramid of images. Larger markers are warped from smaller images. For each marker, the image of the pyramid that minimizes the warping time while preserving the resolution is selected.

respectively. In order to decouple the desired minimum marker size from the input image dimensions, we define $\dot{\tau}_i$ as:

$$\dot{\tau}_i = \tau_c + max(I_w, I_h)\tau_i \mid \tau_i \in [0, 1], \quad (2)$$

where the normalized parameter τ_i indicates the minimum marker size as a value in the range $[0, 1]$. When $\tau_i = 0$, the reduced image will be the same size as the original image. As τ_i tends to one, the image I^r becomes smaller, and consequently, the computational time required for the following step is reduced. The impact of this parameter in the final speed up is measured in the experimental section.

2. *Image Segmentation:* As already indicated, a global threshold method is employed using the following policy. If no markers were detected in the previous frame, a random threshold search is performed. The random process is repeated up to three times using the range of threshold values $[10, 240]$. For each tested threshold value, the whole pipeline explained below is performed. If after a number of attempts, no marker is found, it is assumed that no markers are visible in the frame. If at least one marker is detected, a histogram is created using the pixel values of all detected markers. Then, Otsu's algorithm [40] is employed to select the optimal threshold for the next frame. The calculated threshold is applied to I^r in order to obtain I^t (Fig 2c). As we show experimentally, the proposed method can adapt to smooth and abrupt illumination changes.
3. *Contour Extraction and Filtering:* First, contours are extracted from the image I^t using Suzuki and Abe algorithm [38], then small contours are removed. Since the extracted contours will rarely be squared (due to perspective projection), their perimeter is employed for rejection purposes: those with a perimeter smaller than $P(\tau_c) = 4 \times \tau_c$ pixels are rejected. For the remaining contours, a polygonal approximation is performed using Douglas and Peucker algorithm [39], and those that do not approximate to a convex polygon of four corners are also rejected. Finally, the remaining contours are the candidates to be markers (Fig 2d).
4. *Image Pyramid Creation:* An image pyramid

$$\mathcal{I} = (I^0, \ldots, I^n)$$

with a set of resized versions of I, is created. I^0 denotes the original image and the subsequent images I^i are created by subsampling I^{i-1} by a factor of two. The number n of images in the pyramid is such that the smallest image dimensions is close to $\tau_c \times \tau_c$, i.e.,

$$n = \underset{v|\ I^v \in \mathcal{I}}{\text{argmin}}\ |(I^v_w I^v_h) - \tau_c^2|. \quad (3)$$

5. *Marker Code Extraction:* In this step the canonical images of the remaining contours must be extracted and then binarized. Our method uses the pyramid of images $\mathcal{I}$ previously computed to ensure that the process is performed in constant time, independently of the input image and contour sizes. The key principle is selecting, for each contour, the image from the pyramid in which the contour length is most similar to the canonical image length $P(\tau_c)$. In this manner, warping is faster.

Let us consider a detected contour $\vartheta \in I^r$, and denote by $P(\vartheta)^j$ its perimeter in the image $I^j \in \mathcal{I}$. Then, the best image $I^h \in \mathcal{I}$ for homography computation is selected as:

$$I^h \mid h = \underset{j \in \{0,1,\ldots n\}}{\text{argmin}}\ |P(\vartheta)^j - P(\tau_c)|. \quad (4)$$

The pyramidal warping method employed can be better understood in Fig. 3, which shows a scene with three markers at different distances. The left images represent the canonical images obtained while the right images show the pyramid of images. In our method, the canonical image of the smallest marker is extracted from the largest image in the pyramid (top

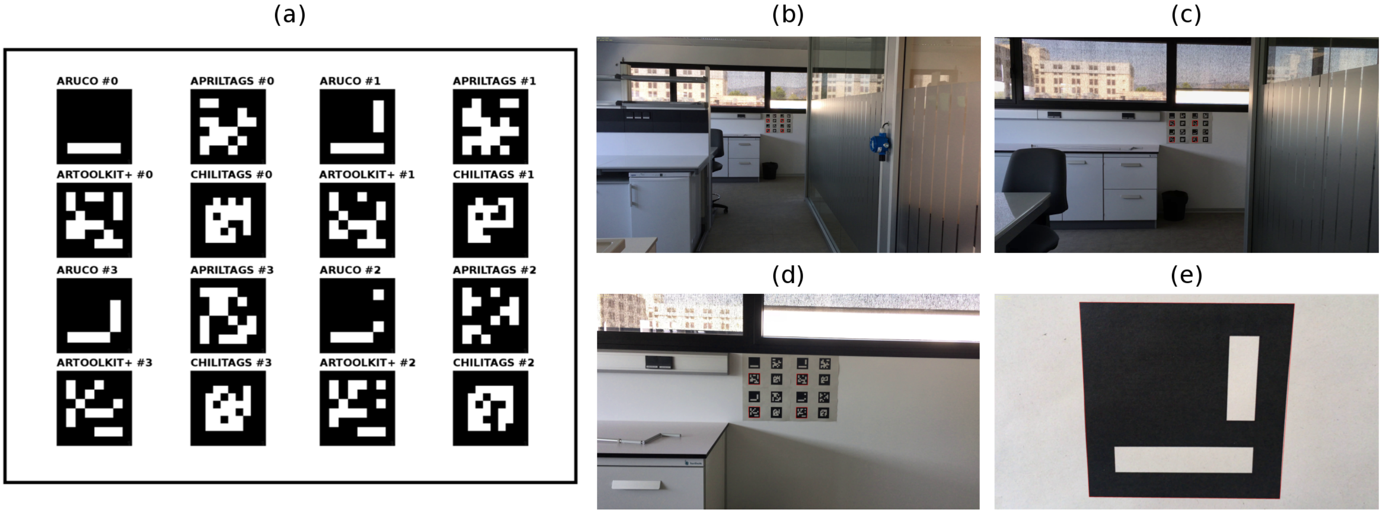


Figure 4: **Test sequences.** (a) The set of 16 markers employed for evaluation. There are four markers from each method tested: ArUco, AprilTags, ArToolKit+ and ChiliTags. (b-e) Images from the video sequences used for testing. The markers are seen as small as in (b), and as big as in (e), where the marker represents the 40% of the total image area.

row of Fig 3). As the length of the marker increases, smaller images of the pyramid are employed to obtain the canonical view. This guarantees that the canonical image is obtained in almost constant time using the minimum possible computation.

Finally, for each canonical image, the Otsu's method [40] for binarization is employed, and the inner code analyzed to determine whether it is a valid marker or not. This is a very cheap operation.

6. *Corner Upsampling:* So far, markers have been detected in the image I^r. However, it is required to precisely localize their corners in the original image I. As previously indicated, the precision of the estimated camera pose is directly influenced by the precision in the corner localization. Since the difference in size between the images I and I^r can be very large, a direct upsampling can lead to errors. Instead, we proceed in incremental steps looking for the corners in larger versions of the image I^r until the image I is reached.

For the corner upsampling task, the image $I^i \in \mathcal{I}$ of the pyramid with the most similar size to I^r is selected in the first place, i.e.,

$$I^i = \underset{I^v \in \mathcal{I}}{\operatorname{argmin}} \; |(I^v_w I^v_h) - (I^r_w I^r_h)|. \tag{5}$$

Then, the position of each contour corner in the image I^i is computed by simply upsampling the corner locations. This is, however, an approximate estimation that does not precisely indicate the corner position in the image I^i. Thus, a corner refinement process is done in the vicinity of each corner so as to find its best location in the selected image I^i. For that purpose, the method implemented in the OpenCV library [41] has been employed. Once the search is done in I^i for all corners, the operation is repeated for the image I^{i-1}, until I^0 is reached. In contrast to the ArUco approach, this one is not affected by lens distortions.

7. *Estimation of τ_i:* The parameter τ_i has a direct influence in the computation time. The higher it is, the faster the computation. A naive approach consists in setting a fixed value for this parameter. However, when processing video sequences, the parameter can be automatically adjusted at the end of each frame. In the first image of the sequence, the parameter τ_i is set to zero. Thus, markers of any size are detected. Then, for the next frame, τ_i is set to a value slightly smaller than the size of the smallest marker detected in the previous frame. In this way, markers could be detected even if the camera moves away from them. Therefore, the parameter τ_i can be dynamically updated as:

$$\tau_i = (1 - \tau_s) P(\vartheta^s)/4 \tag{6}$$

where ϑ^s is the marker with the smallest perimeter found in the image, and τ_s is a factor in the range $(0, 1]$ that accounts for the camera motion speed. For instance, when $\tau_s = 0.1$, it means that in the next frame, τ_i is such that markers 10% smaller than the smallest marker in the current image will be sought. If no markers are detected in a frame, τ_i is set to zero so that in the next frame markers of any size can be detected.

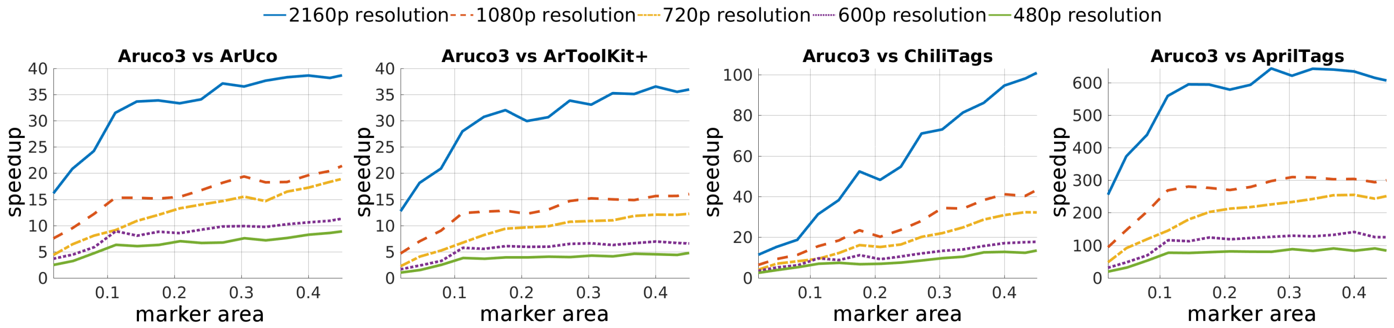


Figure 5: **SpeedUp** of ArUco3 compared to ArUco, ArToolKit+, ChiliTags and AprilTags for resolutions: 4K (3840 × 2160), 1080p (1920 × 1080), 720p (1280 × 720), 600p (800 × 600) and 480p (640 × 480). The horizontal axis represents the percentage of area occupied by the markers in each frame, and the vertical axis one indicates how many times ArUco3 is faster.

As can be observed, the proposed pipeline includes a number of differences with respect to the original ArUco pipeline that allows increasing significantly the processing speed as we show next.

4. Experiments and results

This section shows the results obtained to validate the methodology proposed for the detection of fiducial markers.

First, in Sect 4.1, the computing times of our proposal are compared to the best alternatives found in the literature: AprilTags [18], ChiliTags [36], ArToolKit+ [31], as well as ArUco [17] which is included in the OpenCV library[3]. Then, Sect. 4.2 analyses and compares the sensitivity of the proposed method with the above-mentioned methods. The main goal is to demonstrate that our approach is able to reliably detect the markers with a very high true positive ratio, under a wide range of marker resolutions, while keeping the false positive rate to zero. Afterward, Sect. 4.3 studies the impact of the different system parameters on the speed and sensitivity, while Sect. 4.4 evaluates the precision in the estimation of the corners. Finally, Sect. 4.5 shows the performance of the proposed method in a realistic video sequence with occlusions, illumination, and scale changes.

To carry out the first three experiments, several videos have been recorded in our laboratory. Figure 4(b-e) shows some images of the video sequences employed. For these tests, a panel with a total of 16 markers was printed (Figure 4a), four from each one of the fiducial markers employed. The sequences were recorded at different distances at a frame rate of 30 fps using an Honor 5 mobile phone at 4K resolution. The videos employed are publicly available [4] for evaluation purposes.

In the video, there are frames in which the markers appear as small as can be observed in Figure 4b, where the area of each marker occupies only 0.5% of the image, and frames in which the marker is observed as big as in Figure 4e, where the marker occupies 40% of total image area. In total, the video sequences recorded sum up to 10.666 frames. The video frames have been processed at different resolutions so that the impact of the image resolution in the computing time can be analyzed. In particular, the following the standard image resolutions have been employed: 4K (3840 × 2160), 1080p (1920 × 1080), 720p (1280 × 720), 600p (800 × 600) and 480p (640 × 480).

All tests were performed using an Intel® Core ™ i7-4700HQ 8-core processor with 8 GB RAM and Ubuntu 16.04 as the operating system. However, only one execution thread was employed in the tests performed.

It must be indicated that the code generated as part of this work has been publicly released as the version 3 of the popular ArUco library[5]. So, in the experiments section, the method proposed in this paper will be referred to as ArUco3.

4.1. Speedup

This section compares the computing times of the proposed method with the most commonly used alternatives AprilTags, ArToolKit+, ChiliTags, and ArUco. To do so, we compute the speedup of our approach as the ratio between the computing time of an alternative (t_1) and the computing time of ArUco3 (t_2) in processing the same image:

$$SpeedUp = t_1/t_2 \tag{7}$$

In our method, the value $\tau_c = 32$ was employed in all the sequences, while τ_i and the segmentation threshold where automatically computed as explained in the Steps 2 and 7 of the proposed method (Sect. 3.2).

[3] https://opencv.org/
[4] https://mega.nz/#F!DnA1wIAQ!6f6owb81GOE7Sw3EfddUXQ
[5] http://www.uco.es/grupos/ava/node/25

Table 1: Mean computing times (milliseconds) of the different steps of the proposed method for different resolutions.

	Resolution				
	480p	600p	720p	1080p	2160p
Step 1:Image Resize	0.037	0.050	0.057	0.068	0.101
Step 2:Image Segmentation	0.044	0.048	0.059	0.084	0.351
Step 3:Contour Extraction and Filtering	0.219	0.250	0.301	0.403	1.109
Step 4:Image Pyramid Creation	0.037	0.076	0.096	0.186	0.476
Step 5:Marker code extraction	0.510	0.519	0.542	0.547	0.583
Step 6:Corner Upsampling	0.058	0.065	0.079	0.096	0.134
Time (ms)	0.903	1.009	1.133	1.384	2.755

Fig. 5 shows the speedup of our approach for different image resolutions. The horizontal axis represents the relative area occupied by the marker in the image, while the vertical axis represents the speedup. A total of 30 speed measurements were performed for each image, taking the median computing time for our evaluation. In the tests, the speedup is evaluated as a function of the observed marker area in order to better understand the behavior of our approach.

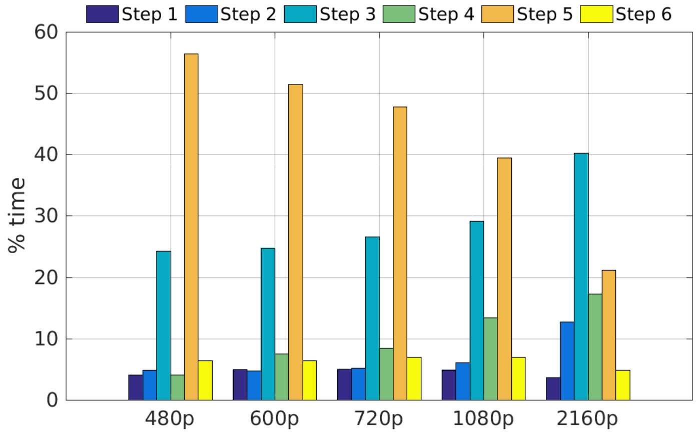
The tests conducted clearly show that the proposed method (ArUco3) is faster than the rest of the methods and that the speedup increases with the image resolution and with the observed marker area. Compared to ArUco implementation in the OpenCV library, the proposed method is significantly faster, achieving a minimum speedup of 17 in $4K$ resolutions, up to 40 in the best case.

In order to properly analyze the computing times of the different steps of the proposed method (Sect. 3.2), Table 1 shows a summary for different image resolutions. Likewise, Fig. 6 shows the percentage of the total time required by each step. Please notice that Step 7 (Eq. 6) has been omitted because its computing time is negligible.

As can be seen, the two most time-consuming operations are Step 3 and 5. In particular, Step 5 requires special attention, since it proves the validity of the multi-scale method proposed for marker warping. It can be observed in the table, that the amount of time employed by Step 5 is constant across all resolutions. In other words, the computing time does not increase significantly with the image resolution. Also notice how the time of Step 3 increases in 2160p. It is because this step involves operations that depend on the image dimensions, which grow quadratically. An interesting future work is to develop methods reducing the time for contour extraction and filtering in high-resolution images.

In any case, considering the average total computing time, the proposed method achieves in average more than 360 fps in $4K$ resolutions and more than 1000 fps in the lowest resolution, without any parallelism.

Figure 6: **Main steps ArUco3 times.** Percentage of time of the global computation required by each of the steps for resolutions: 4K, 1080p, 720p, 600p and 480p.



4.2. Sensitivity analysis

Correct detection of markers is a critical aspect that must be analyzed to verify that the proposed algorithm is able to obviate redundant information present in the scene, extracting exclusively marker information. Fig. 7 shows the True Positive Rate (TPR) of the proposed method as a function of the area occupied by the marker in the image for different image resolutions.

As can be observed, below certain marker area, the detection is not reliable. This is because the observed marker area is very small, making it difficult to distinguish the different bits of the inner binary code. Once the observed area of the marker reaches a certain limit, the proposed method achieves perfect detection in all resolutions. It must be remarked, that the False Positive Rate is zero in all cases tested. Since it is a binary problem, the True Negative Rate is one (TNR=1-FPR).

For a comparative evaluation performance between ArUco3 and the other methods, the TPR has been analyzed individually and the results are shown in Fig. 7. As can be observed, ArUco behaves exactly like ArUco3. AprilTags, however, has very poor behavior in all resolutions, especially as the marker or the image sizes increases. As we already commented in Sect. 2, AprilTags does not rely on warping the marker image but instead does a subsampling of a few pixels on the image in order to obtain the binary code. This may be one of the reasons for its poor performance. ArToolKit+ behaves reasonably well across all the image resolutions and marker areas, while Chilitags shows a somewhat unreliable behavior in all resolutions but 480p.

In conclusion, the proposed approach behaves similar to the previous version of ArUco.

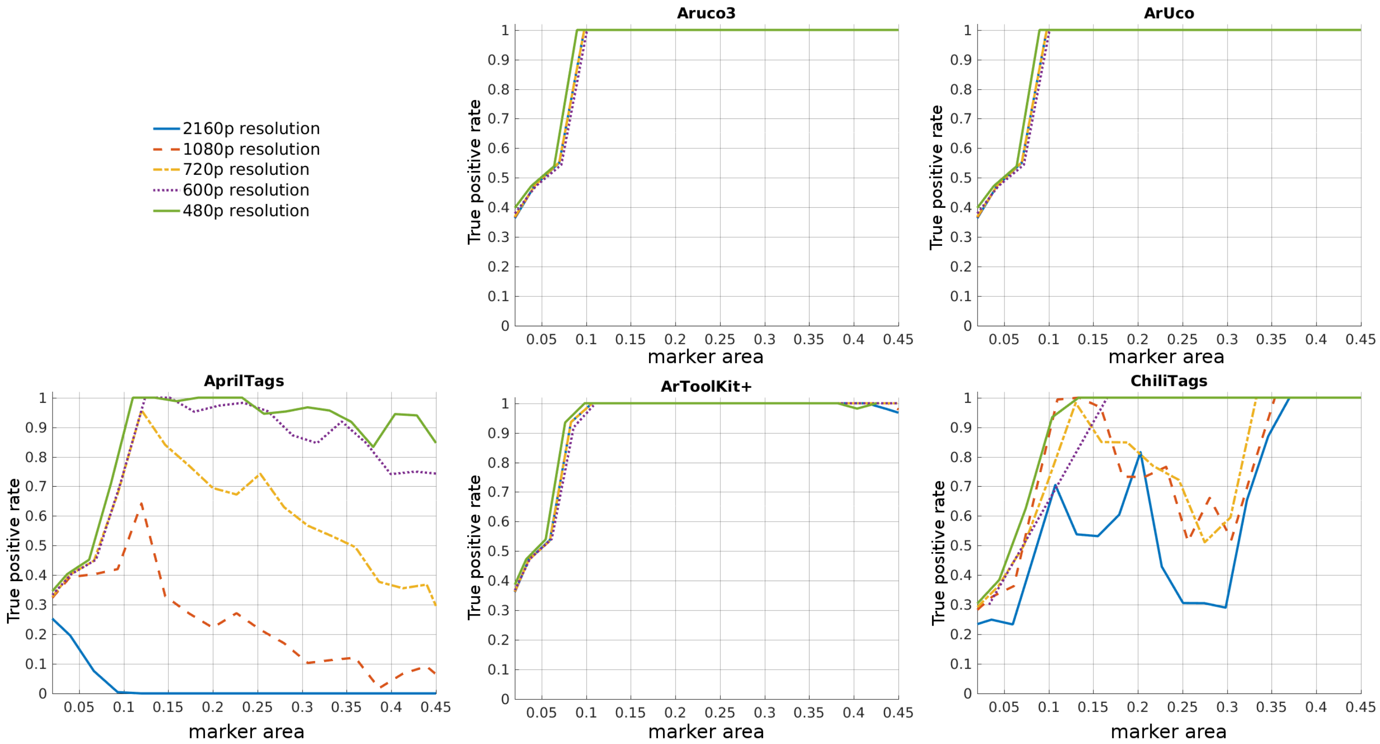


Figure 7: **True Positive Ratio.** Mean true positive ratio (TPR) for ArUco3, Chilitags, ArUco, ArToolKit+ and AprilTags for resolutions: 4K, 1080p, 720p, 600p and 480p), as function of the observed area for the set of markers.

4.3. Analysis of parameters

The computing time and robustness of the proposed method depend mainly on two parameters, namely τ_i which indicates the minimum size of the markers detected, and τ_c, the size of the canonical image.

The parameter τ_i has an influence on the computing time, since it determines the size of the resized image I^r (Eq. 1). We have analyzed the speed as a function of this parameter and the results are shown in Fig. 8. The figure represents the horizontal axis the value τ_i, and in the vertical axis, the average speed (measured as frames per second) in the sequences analyzed, independently of the observed marker area. A different line has been depicted for each image resolution. In this case, we have set fixed the parameter $\tau_c = 32$.

It can be observed that the curves follow a similar pattern in the five cases analyzed. In general, the maximum increase in speed is obtained in the range of values $\tau_i = (0, 0.2)$. Beyond that point, the improvement becomes marginal. To better understand the impact of this parameter, Table 2 shows the reduction of the input image size I for different values of τ_i. For instance, when $\tau_i = 0.02$, the resized image I^r is 48% smaller than the original input image I (see Eq. 1). Beyond $\tau_i = 0.2$, the resized image is so small that it has not a big impact in the speedup because there are other steps with a fixed computing time such as the Step 5 (Marker Code Extraction).

Table 2: Image size reduction for different values of τ_i.

τ_i	0.01	0.015	0.02	0.1	0.2
Size reduction	0%	31%	48%	82%	90%

In any case, it must be noticed that the proposed method is able to achieve 1000 fps in 4K resolutions when detecting markers larger than 10% ($\tau_i = 0.1$) of the image area, and the same limit of 1000 fps is achieved for 1080p resolutions for $\tau_i = 0.05$.

With regards to the parameter τ_c, it indirectly influences the speed since it determines the size of the resized images (Eq 1). The smaller it is, the smaller the resized image I^r. Nevertheless, this parameter also has an influence on the correct detection of the markers. The parameter indicates the size of the canonical images used to identify the binary code of markers. If the canonical image is very small, pixels are mixed up, and identification is not robust. Consequently, the goal is to determine the minimum value of τ_c that achieves the best TPR. Fig. 9 shows the TPR obtained for different configurations of the parameter τ_c. As can be seen, for low values of the parameter τ_c (between 8 and 32) the system shows problems in the detection of markers. However, for $\tau_c \geq 32$ there is no improvement in the TPR. Thus, we conclude that the value $\tau_c = 32$ is the

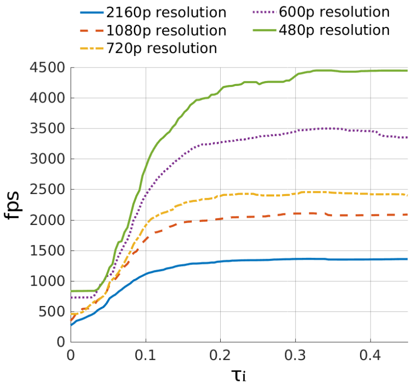


Figure 8: **Parameter τ_i.** Speed of method as a function of the parameter τ_i for the different resolutions tested.

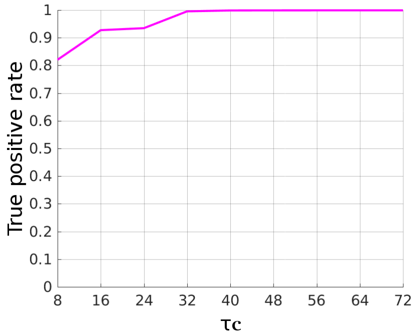


Figure 9: **Parameter τ_c.** True positive rate obtained by different configurations of parameter τ_c

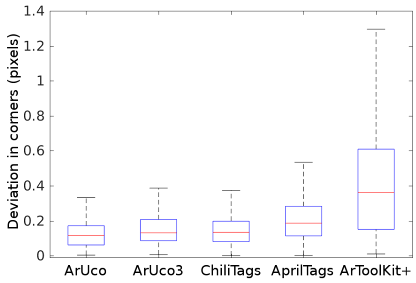


Figure 10: **Vertex jitter** measured for the different marker systems.

best choice.

4.4. Precision of corner detection

An important aspect to consider in the detection of the markers is *vertex jitter*, which refers to the noise in the estimation of the corners' location. These errors are problematic because they propagate to the estimation of the camera pose. In our method, a corner upsampling step (Step 6 in Sect. 3.2) is proposed to refine the corners' estimations from the reduced image I^r to the original image I. This section analyzes the proposed method comparing the results with the other marker systems.

In order to perform the experiments, the camera has been placed at a fixed position recording the set of markers already presented in Fig. 4a. Since the camera is not moving, the average location estimated for each corner can be considered to be the correct one (i.e., a Gaussian error distribution is assumed). Then, the standard deviation is an error measure for the localization of the corners. The process has been repeated a total of six times at varying distances and the results obtained are shown in Fig. 10 as box plots. In Table 3, the average error of each method has been indicated.

Table 3: Vertex jitter analysis: Standard deviations of the different methods in estimating the marker corners.

Method	ArUco	ArUco3	Chilitags	AprilTags	ArToolKit+
Average error (pix)	0.140	0.161	0.174	0.225	0.432

As can be observed, the ArUco system obtains the best results, followed by our proposal ArUco3. However, it can be seen that the difference between both methods is of only 0.02 pixels, which is very small to consider it relevant. Chilitags shows a similar behavior than ArUco and ArUco3, but AprilTags and ArToolKit+ exhibit worse performance.

4.5. Video sequence analysis

This section aims at showing the behavior of the proposed system in a realistic scenario. For that purpose, four markers have been placed in an environment with irregular lighting and a video sequence has been recorded using a 4K mobile phone camera. Figure 11(a-e) show the frames $1, 665, 1300, 1700$ and 2100 of the video sequence. At the start of the sequence, the camera is around five meters away from the markers. The camera approaches the markers and then moves away again. As can be seen, around frame 650 (Figure 11b), the user occludes the markers temporarily.

Figure 11f shows the values of the parameter τ_i automatically calculated along the sequence and Figure 11g the processing speed. As can be observed, the system is able to automatically adapt the value of τ_i according to the observed marker area, thus adapting the computing speed of the system. The maximum speed is obtained around the frame 1300 when the camera is closest to the markers.

It can also be observed that around frame 650 when the user occludes the markers with his hand, the system is unable to detect any marker. Thus, the system searches for the full resolution image ($\tau_i = 0$) and the speed decreases. However, when the markers are observed again, the system recovers its speed.

Finally, Figure 11h shows the threshold values employed for segmentation in each frame. As can be seen, the system adapts to the illumination changes. Along the sequence, the system does not produce any false negative nor positives.

5. Conclusions and future work

This paper has proposed a novel approach for detecting fiducial markers aimed at maximizing speed while preserving accuracy and robustness. The proposed method

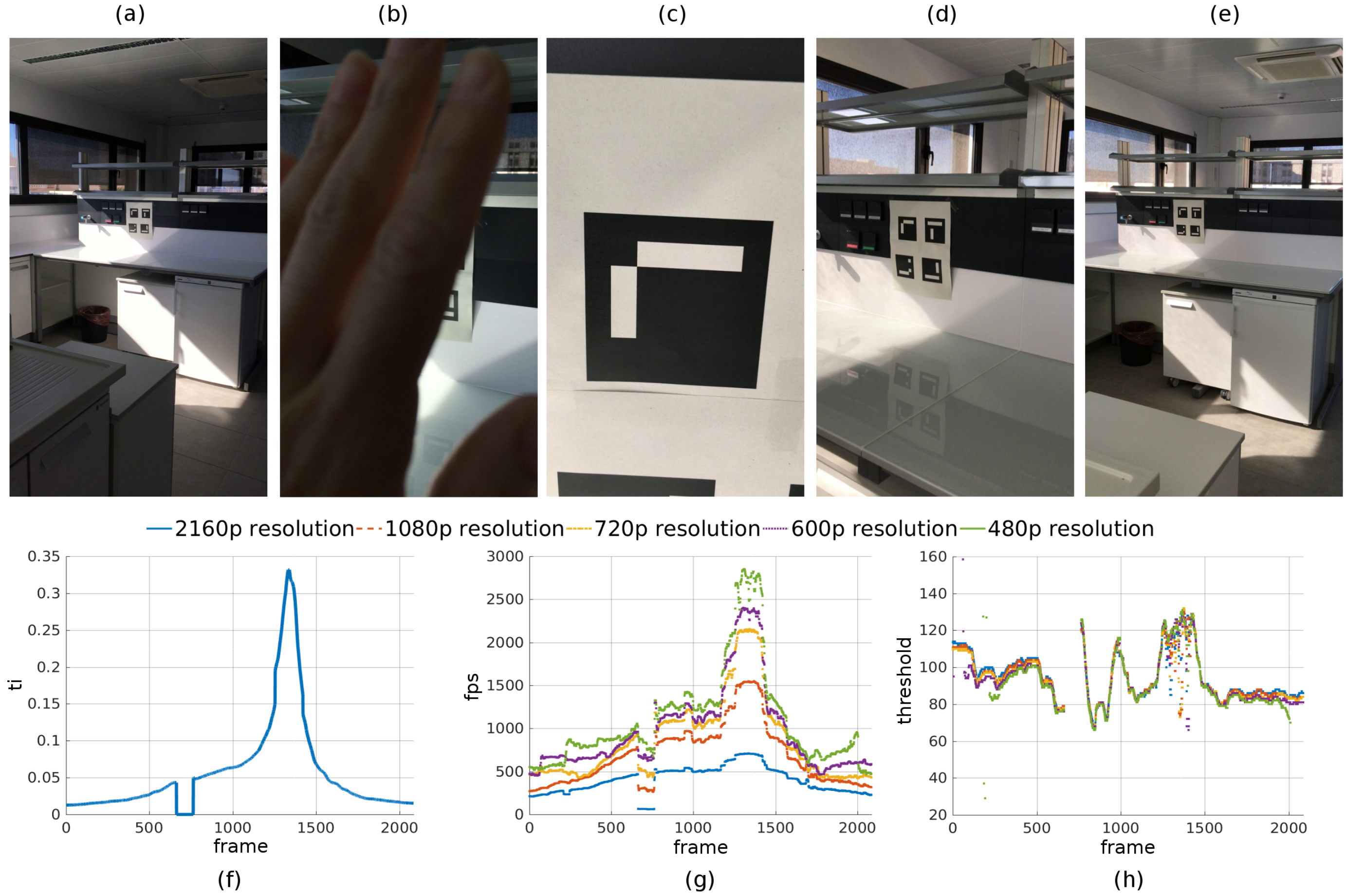


Figure 11: **Video Sequence** in a realistic scenario. (a-e) Frames of the video sequence. The camera approaches the marker and then moves away. The user occludes the camera temporarily. (f) Evolution of the parameter τ_i automatically computed. (g) Speed of the proposed method in each frame of the sequence. (h) Thresholds automatically computed for each frame. The system adapts to illumination changes.

is specially designed to take advantage of the increasing camera resolutions available nowadays. Instead of detecting markers in the original image, a smaller version of the image is employed, in which the detection can be done at higher speed. By wisely employing a multi-scale image representation, the proposed method is able to find the position of the marker corners with subpixel accuracy in the original image. The size of the processed image, as well as the threshold employed for segmentation, are dynamically adapted in each frame considering the information of the previous one. As a consequence, the system speed dynamically adapts in order to achieve the maximum performance.

As shown experimentally, the proposed method outperforms the state-of-the-art systems in terms of computing speed, without compromising the sensitivity or the precision. Our method is between 17 and 40 times faster than the ArUco approach implemented in the OpenCV library. When compared to other approaches such as Chilitags, AprilTags, and ArToolKit+, our method achieves even higher speedups.

We consider as possible future works to investigate the use of the proposed method in fish-eye cameras. The idea is to compare the method with the rectified images if there is analyze the method's performance in presence of high distortion. Also, we as well as to characterize the performance when multiple fiducial markers with significantly different scales are present in the same image.

Our system, which is publicly available as open source code[6], is a cost-effective tool for fast and precise self-localization in applications such as robotics, unmanned vehicles or augmented reality applications.

Acknowledgments

This project has been funded under projects TIN2016-75279-P and IFI16/00033 (ISCIII) of Spain Ministry of Economy, Industry and Competitiveness, and FEDER.

[6]`http://www.uco.es/grupos/ava/node/25`

References

[1] R. Sim, J. J. Little, Autonomous vision-based robotic exploration and mapping using hybrid maps and particle filters, Image and Vision Computing 27 (1) (2009) 167 – 177, canadian Robotic Vision 2005 and 2006.

[2] A. Pichler, S. C. Akkaladevi, M. Ikeda, M. Hofmann, M. Plasch, C. Wögerer, G. Fritz, Towards shared autonomy for robotic tasks in manufacturing, Procedia Manufacturing 11 (Supplement C) (2017) 72 – 82, 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy.

[3] R. Valencia-Garcia, R. Martinez-Béjar, A. Gasparetto, An intelligent framework for simulating robot-assisted surgical operations, Expert Systems with Applications 28 (3) (2005) 425 – 433.

[4] A. Broggi, E. Dickmanns, Applications of computer vision to intelligent vehicles, Image and Vision Computing 18 (5) (2000) 365 – 366.

[5] T. Patterson, S. McClean, P. Morrow, G. Parr, C. Luo, Timely autonomous identification of uav safe landing zones, Image and Vision Computing 32 (9) (2014) 568 – 578.

[6] D. González, J. Pérez, V. Milanés, Parametric-based path generation for automated vehicles at roundabouts, Expert Systems with Applications 71 (2017) 332 – 341.

[7] J. L. Sanchez-Lopez, J. Pestana, P. de la Puente, P. Campoy, A reliable open-source system architecture for the fast designing and prototyping of autonomous multi-uav systems: Simulation and experimentation, Journal of Intelligent & Robotic Systems (2015) 1–19.

[8] M. Olivares-Mendez, S. Kannan, H. Voos, Vision based fuzzy control autonomous landing with uavs: From v-rep to real experiments, in: Control and Automation (MED), 2015 23th Mediterranean Conference on, 2015, pp. 14–21.

[9] S. Pflugi, R. Vasireddy, T. Lerch, T. M. Ecker, M. Tannast, N. Boemke, K. Siebenrock, G. Zheng, Augmented marker tracking for peri-acetabular osteotomy surgery, in: 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2017, pp. 937–941.

[10] J. P. Lima, R. Roberto, F. Simões, M. Almeida, L. Figueiredo, J. M. Teixeira, V. Teichrieb, Markerless tracking system for augmented reality in the automotive industry, Expert Systems with Applications 82 (2017) 100 – 114.

[11] P. Chen, Z. Peng, D. Li, L. Yang, An improved augmented reality system based on andar, Journal of Visual Communication and Image Representation 37 (2016) 63 – 69, weakly supervised learning and its applications.

[12] S. Khattak, B. Cowan, I. Chepurna, A. Hogue, A real-time reconstructed 3d environment augmented with virtual objects rendered with correct occlusion, in: Games Media Entertainment (GEM), 2014 IEEE, 2014, pp. 1–8.

[13] J. Engel, T. Schöps, D. Cremers, LSD-SLAM: Large-scale direct monocular SLAM, 2014.

[14] R. Mur-Artal, J. M. M. Montiel, J. D. Tardós, Orb-slam: A versatile and accurate monocular slam system, IEEE Transactions on Robotics 31 (5) (2015) 1147–1163.

[15] Cooperative pose estimation of a fleet of robots based on interactive points alignment, Expert Systems with Applications 45 (2016) 150 – 160.

[16] S.-h. Zhong, Y. Liu, Q.-c. Chen, Visual orientation inhomogeneity based scale-invariant feature transform, Expert Syst. Appl. 42 (13) (2015) 5658–5667.

[17] S. Garrido-Jurado, R. Muñoz Salinas, F. J. Madrid-Cuevas, M. J. Marín-Jiménez, Automatic generation and detection of highly reliable fiducial markers under occlusion, Pattern Recognition 47 (6) (2014) 2280–2292.

[18] E. Olson, Apriltag: A robust and flexible visual fiducial system, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, 2011, pp. 3400–3407.

[19] F. Ababsa, M. Mallem, Robust camera pose estimation using 2d fiducials tracking for real-time augmented reality systems, in: Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry, VRCAI '04, 2004, pp. 431–435.

[20] V. Mondéjar-Guerra, S. Garrido-Jurado, R. Muñoz-Salinas, M.-J. Marín-Jiménez, R. Medina-Carnicer, Robust identification of fiducial markers in challenging conditions, Expert Systems with Applications 93 (1) (2018) 336–345.

[21] R. Muñoz-Salinas, M. J. Marín-Jimenez, E. Yeguas-Bolivar, R. Medina-Carnicer, Mapping and localization from planar markers, Pattern Recognition 73 (January 2018) 158 – 171.

[22] K. Dorfmüller, H. Wirth, Real-time hand and head tracking for virtual environments using infrared beacons, in: in Proceedings CAPTECH'98. 1998, Springer, 1998, pp. 113–127.

[23] M. Ribo, A. Pinz, A. L. Fuhrmann, A new optical tracking system for virtual and augmented reality applications, in: In Proceedings of the IEEE Instrumentation and Measurement Technical Conference, 2001, pp. 1932–1936.

[24] V. A. Knyaz, R. V. Sibiryakov, The development of new coded targets for automated point identification and non-contact surface measurements, in: 3D Surface Measurements, International Archives of Photogrammetry and Remote Sensing, Vol. XXXII, part 5, 1998, pp. 80–85.

[25] L. Naimark, E. Foxlin, Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker, in: Proceedings of the 1st International Symposium on Mixed and Augmented Reality, ISMAR '02, IEEE Computer Society, Washington, DC, USA, 2002, pp. 27–36.

[26] J. Rekimoto, Y. Ayatsuka, Cybercode: designing augmented reality environments with visual tags, in: Proceedings of DARE 2000 on Designing augmented reality environments, DARE '00, ACM, New York, NY, USA, 2000, pp. 1–10.

[27] M. Rohs, B. Gfeller, Using camera-equipped mobile phones for interacting with real-world objects, in: Advances in Pervasive Computing, 2004, pp. 265–271.

[28] M. Kaltenbrunner, R. Bencina, reactivision: a computer-vision framework for table-based tangible interaction, in: Proceedings of the 1st international conference on Tangible and embedded interaction, TEI '07, ACM, New York, NY, USA, 2007, pp. 69–74.

[29] H. Kato, M. Billinghurst, Marker tracking and hmd calibration for a video-based augmented reality conferencing system, in: Augmented Reality, 1999. (IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop on, 1999, pp. 85–94.

[30] S. Lin, D. J. Costello, Error Control Coding, Second Edition, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.

[31] D. Wagner, D. Schmalstieg, ARToolKitPlus for pose tracking on mobile devices, in: Computer Vision Winter Workshop, 2007, pp. 139–146.

[32] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavári, L. M. Encarnação, M. Gervautz, W. Purgathofer, The studierstube augmented reality project, Presence: Teleoper. Virtual Environ. 11 (1) (2002) 33–54.

[33] M. Fiala, Designing highly reliable fiducial markers, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (7) (2010) 1317–1324.

[34] D. Flohr, J. Fischer, A Lightweight ID-Based Extension for Marker Tracking Systems, in: Eurographics Symposium on Virtual Environments (EGVE) Short Paper Proceedings, 2007, pp.

59–64.
[35] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, R. Medina-Carnicer, Generation of fiducial marker dictionaries using mixed integer linear programming, Pattern Recognition 51 (2016) 481–491.
[36] Q. Bonnard, S. Lemaignan, G. Zufferey, A. Mazzei, S. Cuendet, N. Li, A. Özgür, P. Dillenbourg, Chilitags 2: Robust fiducial markers for augmented reality and robotics. (2013).
URL `http://chili.epfl.ch/software`
[37] D. Johnston, M. Fleury, A. Downton, A. Clark, Real-time positioning for augmented reality on a custom parallel machine, Image and Vision Computing 23 (3) (2005) 271 – 286.
[38] Topological structural analysis of digitized binary images by border following, Computer Vision, Graphics, and Image Processing 30 (1) (1985) 32 – 46.
[39] D. H. Douglas, T. K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, Cartographica: The International Journal for Geographic Information and Geovisualization 2 (10) (1973) 112 – 122.
[40] N. Otsu, A threshold selection method from gray-level histograms, IEEE Transactions on Systems, Man, and Cybernetics 9 (1) (1979) 62–66.
[41] G. Bradski, A. Kaehler, Learning OpenCV: Computer Vision in C++ with the OpenCV Library, 2nd Edition, O'Reilly Media, Inc., 2013.